skrypt do pobrania:

python_hyper-v_topology_grapher.py

PY: Topology mapper Hyper-V

Wprowadzenie

Poniższy kod umożliwia pobieranie informacji o przełącznikach wirtualnych (VMSwitches) oraz adapterach sieciowych maszyn wirtualnych (VMNetworkAdapters) w systemie Hyper-V. Następnie buduje on graf, gdzie węzły to przełączniki wirtualne oraz maszyny wirtualne, a krawędzie reprezentują połączenia między tymi elementami. Całość jest wizualizowana przy użyciu biblioteki Matplotlib.

Zawartość Programu

Program składa się z kilku funkcji, które realizują różne zadania:

1. Funkcja get_vmswitches

```
def get_vmswitches():
    """
    Use PowerShell to get Hyper-V virtual switches as JSON.
    """
    cmd = ["powershell", "-Command", "Get-VMSwitch | ConvertTo-Json"]
    proc = subprocess.run(cmd, capture_output=True, text=True)
    try:
        switches = json.loads(proc.stdout)
    except json.JSONDecodeError as e:
        print("Error decoding JSON for switches:", e)
        print("PowerShell output was:", proc.stdout)
        switches = []
    # Ensure the data is a list
    if isinstance(switches, dict):
        switches = [switches]
    return switches
```

Funkcja ta używa PowerShella do pobrania informacji o przełącznikach wirtualnych w systemie Hyper-V, a następnie konwertuje wynik na format JSON. Jeśli dekodowanie JSON zakończy się niepowodzeniem, wypisuje komunikat o błędzie i zwraca pustą listę. Jeśli dane nie są w formie listy, funkcja konwertuje je na listę.

2. Funkcja get_vmnetworkadapters

```
def get_vmnetworkadapters():
    """
    Use PowerShell to get Hyper-V VM network adapters as JSON.
    This version pipes all VMs to Get-VMNetworkAdapter to ensure we get
data.
    """
    # Updated command to get network adapters from all VMs
    cmd = ["powershell", "-Command", "Get-VM | Get-VMNetworkAdapter |
ConvertTo-Json"]
    proc = subprocess.run(cmd, capture_output=True, text=True)
    if proc.stderr:
        print("PowerShell error output:", proc.stderr)
    try:
        adapters = json.loads(proc.stdout)
    except json.JSONDecodeError as e:
        print("Error decoding JSON for adapters:", e)
        print("PowerShell output was:", proc.stdout)
        adapters = []
    # Ensure the data is a list
    if isinstance(adapters, dict):
        adapters = [adapters]
    return adapters
```

Ta funkcja działa podobnie do poprzedniej, ale zamiast przełączników wirtualnych, pobiera dane o adapterach sieciowych maszyn wirtualnych w Hyper-V. W przypadku błędów podczas dekodowania JSON, wypisuje komunikat o błędzie i zwraca pustą listę.

3. Funkcja build_graph

```
def build_graph(switches, adapters):
    """
    Build a graph with virtual switches as one type of node and VMs as
another.
    An edge represents a connection between a VM and a virtual switch.
    """
    G = nx.Graph()

    # Add virtual switch nodes
    for sw in switches:
        sw_name = sw.get("Name")
        if sw_name:
            G.add_node(sw_name, type="switch")

    # Add VM nodes and edges based on network adapter connections
    for adapter in adapters:
        vm_name = adapter.get("VMName")
        switch_name = adapter.get("SwitchName")
```

```
        if vm_name:
            G.add_node(vm_name, type="vm")
            # Create an edge if the adapter is connected to a switch
            if switch_name:
                G.add_edge(vm_name, switch_name)
    return G
```

Funkcja ta tworzy graf przy użyciu biblioteki NetworkX. Dodaje węzły do grafu reprezentujące przełączniki wirtualne (o typie „switch”) oraz maszyny wirtualne (o typie „vm”). Krawędzie grafu tworzone są pomiędzy maszynami wirtualnymi a przełącznikami wirtualnymi, jeśli adapter sieciowy jest podłączony do danego przełącznika.

4. Funkcja draw_graph

```
def draw_graph(G):
    """
    Draw the graph using matplotlib.
    Switches are drawn as squares (blue) and VMs as circles (green).
    """
    pos = nx.spring_layout(G, k=1.0, iterations=100, seed=42)  # For
consistent layout

    # Separate nodes by type for custom styling
    switch_nodes = [n for n, d in G.nodes(data=True) if d.get("type") ==
"switch"]
    vm_nodes = [n for n, d in G.nodes(data=True) if d.get("type") == "vm"]

    plt.figure(figsize=(10, 8))
    nx.draw_networkx_nodes(G, pos, nodelist=switch_nodes,
node_color='lightblue', node_shape='s',
                           node_size=1500, label="Virtual Switch")
    nx.draw_networkx_nodes(G, pos, nodelist=vm_nodes,
node_color='lightgreen', node_shape='o',
                           node_size=1500, label="Virtual Machine")
    nx.draw_networkx_edges(G, pos)
    nx.draw_networkx_labels(G, pos, font_size=10)

    plt.title("Hyper-V Topology: Virtual Switches & VMs")
    plt.axis('off')
    plt.legend(scatterpoints=1, labelspacing=1.5, handletextpad=1.0,
borderaxespad=1.0)
    plt.tight_layout()
    plt.show()
```

Funkcja ta odpowiada za wizualizację grafu za pomocą biblioteki Matplotlib. Węzły reprezentujące przełączniki wirtualne są rysowane jako niebieskie kwadraty, a węzły reprezentujące maszyny wirtualne jako zielone okręgi. Krawędzie grafu reprezentują połączenia między przełącznikami a maszynami wirtualnymi.

5. Funkcja main

```
def main():
    print("Extracting Hyper-V virtual switches...")
    switches = get_vmswitches()
    print(f"Found {len(switches)} switch(es).")

    print("Extracting Hyper-V VM network adapters...")
    adapters = get_vmnetworkadapters()
    print(f"Found {len(adapters)} network adapter(s).")

    if not switches and not adapters:
        print("No data retrieved. Ensure Hyper-V is installed and you have
proper permissions.")
        return

    graph = build_graph(switches, adapters)
    draw_graph(graph)
```

Funkcja `main` uruchamia proces zbierania danych o przełącznikach wirtualnych i adapterach sieciowych maszyn wirtualnych, a następnie buduje i rysuje graf przedstawiający te dane. Funkcja ta jest punktem wejścia programu i jest uruchamiana, gdy program jest wykonywany bezpośrednio.

6. Wywołanie programu

```
if __name__ == "__main__":
    main()
```

Jest to standardowy sposób uruchamiania programu w Pythonie. Jeżeli skrypt jest uruchamiany bezpośrednio, wywoływana jest funkcja `main()`, która rozpoczyna cały proces.

Podsumowanie

Kod służy do pobierania danych o przełącznikach wirtualnych i adapterach sieciowych maszyn wirtualnych w systemie Hyper-V, budowania grafu przedstawiającego te elementy, a następnie wizualizowania tego grafu za pomocą biblioteki Matplotlib. Używa on PowerShella do pozyskiwania danych oraz NetworkX do manipulacji grafem.